Clustering Mixtures with Almost Optimal Separation in Polynomial Time

Allen Liu (MIT)

Joint work with Jerry Li (Microsoft Research)

(Gaussian) Mixture Models

(Gaussian) Mixture Models

Mixture Models

Given a class of distributions $\mathcal{D}$, a mixture of k elements from $\mathcal{D}$ is a distribution of the form

$$\mathcal{M} = \sum_{i=1}^{k} w_i D_i \ ,$$

where $D_1, \ldots, D_k \in \mathcal{D}$, and w_i satisfy $w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$.

(Gaussian) Mixture Models

Mixture Models

Given a class of distributions $\mathcal{D}$, a mixture of k elements from $\mathcal{D}$ is a distribution of the form

$$\mathcal{M} = \sum_{i=1}^{k} w_i D_i \ ,$$

where $D_1, \ldots, D_k \in \mathcal{D}$, and w_i satisfy $w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$.

Gaussian Mixture Models (GMMs): $\mathcal{D} = \{N(\mu, \Sigma)\}$.

(Gaussian) Mixture Models

Mixture Models

Given a class of distributions $\mathcal{D}$, a mixture of k elements from $\mathcal{D}$ is a distribution of the form

$$\mathcal{M} = \sum_{i=1}^{k} w_i D_i \ ,$$

where $D_1, \ldots, D_k \in \mathcal{D}$, and w_i satisfy $w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$.

Gaussian Mixture Models (GMMs): $\mathcal{D} = \{N(\mu, \Sigma)\}$.

When $\Sigma = I$, these are known as *isotropic GMMs*

(Gaussian) Mixture Models

Mixture Models

Given a class of distributions $\mathcal{D}$, a mixture of k elements from $\mathcal{D}$ is a distribution of the form

$$\mathcal{M} = \sum_{i=1}^{k} w_i D_i \ ,$$

where $D_1, \ldots, D_k \in \mathcal{D}$, and w_i satisfy $w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$.

Gaussian Mixture Models (GMMs): $\mathcal{D} = \{N(\mu, \Sigma)\}$.

When $\Sigma = I$, these are known as *isotropic GMMs*

Mixture models and GMMs are well-studied theoretically, and popular in practice as a way to model heterogeneous data.

Clustering mixture models

Clustering mixture models

Given samples $X_1, \ldots, X_n$ from a GMM (or any mixture model), can we *cluster* the samples, i.e. group the samples that came from the same components together?

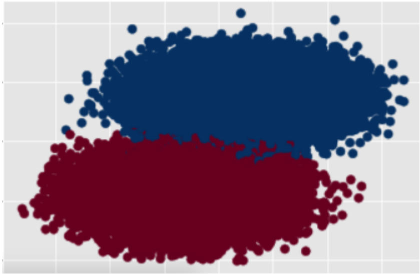
Clustering mixture models

Given samples $X_1, \ldots, X_n$ from a GMM (or any mixture model), can we *cluster* the samples, i.e. group the samples that came from the same components together?

Separation conditions

Separation conditions

Need separation in TV-distance between components for clustering to be possible information-theoretically



Separation Conditions

Separation Conditions

For isotropic GMMs, we need component means to be separated

Separation Conditions

For isotropic GMMs, we need component means to be separated

Let $\mathcal{M} = \sum_{i=1}^{k} w_i N(\mu_i, I)$ be a mixture of k isotropic Gaussians, and define

$$\Delta = \min_{i \neq j} \|\mu_i - \mu_j\|_2 \ .$$

Separation Conditions

For isotropic GMMs, we need component means to be separated

Let $\mathcal{M} = \sum_{i=1}^{k} w_i N(\mu_i, I)$ be a mixture of k isotropic Gaussians, and define

$$\Delta = \min_{i \neq j} \|\mu_i - \mu_j\|_2 \ .$$

Main Question

What is the minimum Δ you need to *efficiently* cluster?

The information theoretic limit

The information theoretic limit

For simplicity, assume that mixing weights are uniform i.e. $w_i = 1/k$ for all $i = 1, \ldots, k$.

The results do not qualitatively change for general mixing weights.

The information theoretic limit

For simplicity, assume that mixing weights are uniform i.e. $w_i = 1/k$ for all $i = 1, \ldots, k$.

The results do not qualitatively change for general mixing weights.

Fact [Regev, Vijayaraghavan 2017]

$\Delta = \Theta(\sqrt{\log k})$ is both necessary and sufficient to obtain a clustering that is 99% accurate with high probability.

What about computationally efficient methods?

What about computationally efficient methods?

Clustering is easy in 1-dimension.

What about computationally efficient methods?

Clustering is easy in 1-dimension.

In high dimensions, we can brute-force search in $\exp(d)$ time (where d is the dimensionality of the data). What can we achieve with efficient methods?

What about computationally efficient methods?

Clustering is easy in 1-dimension.

In high dimensions, we can brute-force search in $\exp(d)$ time (where d is the dimensionality of the data). What can we achieve with efficient methods?

[Dasgupta, 1999]: $\Delta = \Omega(d^{1/2})$ in time $\mathrm{poly}(d, k)$.

What about computationally efficient methods?

Clustering is easy in 1-dimension.

In high dimensions, we can brute-force search in $\exp(d)$ time (where d is the dimensionality of the data). What can we achieve with efficient methods?

[Dasgupta, 1999]: $\Delta = \Omega(d^{1/2})$ in time $\mathrm{poly}(d, k)$.
[Vempala, Wang 2004]: $\Delta = \Omega(\min(d^{1/4}, k^{1/4}))$ in time $\mathrm{poly}(d, k)$.

What about computationally efficient methods?

Clustering is easy in 1-dimension.

In high dimensions, we can brute-force search in $\exp(d)$ time (where d is the dimensionality of the data). What can we achieve with efficient methods?

[Dasgupta, 1999]: $\Delta = \Omega(d^{1/2})$ in time $\mathrm{poly}(d, k)$.
[Vempala, Wang 2004]: $\Delta = \Omega(\min(d^{1/4}, k^{1/4}))$ in time $\mathrm{poly}(d, k)$.

[Diakonikolas, Kane, Stewart 2018], [Kothari, Steinhardt, Steurer 2018], [Hopkins, Li 2018]

- All get $\Delta = \Omega(k^{\epsilon})$ in time $\mathrm{poly}(d, k^{\mathrm{poly}(1/\epsilon)})$

What about computationally efficient methods?

Clustering is easy in 1-dimension.

In high dimensions, we can brute-force search in $\exp(d)$ time (where d is the dimensionality of the data). What can we achieve with efficient methods?

[Dasgupta, 1999]: $\Delta = \Omega(d^{1/2})$ in time $\mathrm{poly}(d, k)$.
[Vempala, Wang 2004]: $\Delta = \Omega(\min(d^{1/4}, k^{1/4}))$ in time $\mathrm{poly}(d, k)$.

[Diakonikolas, Kane, Stewart 2018], [Kothari, Steinhardt, Steurer 2018], [Hopkins, Li 2018]

- All get $\Delta = \Omega(k^{\epsilon})$ in time $\mathrm{poly}(d, k^{\mathrm{poly}(1/\epsilon)})$

Question

Can we cluster in polynomial time down to the information theoretic limit?

Main Result

Main Result

Theorem

Let $c > 0$, and let $\mathcal{M}$ be a (uniform) mixture of isotropic Gaussians with separation $\Delta = \Omega(\log^{1/2+c} k)$. Then, there is an algorithm which takes $n = \mathrm{poly}(k, d)$ samples from $\mathcal{M}$ and runs in time $\mathrm{poly}(k, d)$, and which recovers a perfect clustering of the samples with high probability.

Main Result

Theorem

Let $c > 0$, and let $\mathcal{M}$ be a (uniform) mixture of isotropic Gaussians with separation $\Delta = \Omega(\log^{1/2+c} k)$. Then, there is an algorithm which takes $n = \mathrm{poly}(k, d)$ samples from $\mathcal{M}$ and runs in time $\mathrm{poly}(k, d)$, and which recovers a perfect clustering of the samples with high probability.

- Our algorithm also works for non-uniform mixtures.

Main Result

Theorem

Let $c > 0$, and let $\mathcal{M}$ be a (uniform) mixture of isotropic Gaussians with separation $\Delta = \Omega(\log^{1/2+c} k)$. Then, there is an algorithm which takes $n = \mathrm{poly}(k, d)$ samples from $\mathcal{M}$ and runs in time $\mathrm{poly}(k, d)$, and which recovers a perfect clustering of the samples with high probability.

- Our algorithm also works for non-uniform mixtures.
- We can also handle mixtures of shifts of any distribution D satisfying the Poincaré inequality under a mild additional condition

Outline

- **The barrier to existing approaches**
- Our techniques
  - Implicit Moment Estimation
  - Implicit Moment Computation
- Putting it all together

Method of Moments

Method of Moments

At a high-level:

1. Measure moments of the mixture $\mathcal{M}$ i.e. $\mathbb{E}_{X\sim\mathcal{M}}[X^{\otimes t}]$

Method of Moments

At a high-level:

1. Measure moments of the mixture $\mathcal{M}$ i.e. $\mathbb{E}_{X \sim \mathcal{M}}[X^{\otimes t}]$
2. If moments are distorted compared to those of a standard Gaussian then we can cluster

Method of Moments

At a high-level:

1. Measure moments of the mixture $\mathcal{M}$ i.e. $\mathbb{E}_{X\sim\mathcal{M}}[X^{\otimes t}]$
2. If moments are distorted compared to those of a standard Gaussian then we can cluster
3. If separation between means is $\Omega(k^{\epsilon})$ then we need to measure moments of degree $1/\epsilon$ to detect distortions

The barrier to reaching polylogarithmic separation

The barrier to reaching polylogarithmic separation

Barrier: Clustering with separation $\Omega(k^\epsilon)$ requires degree $1/\epsilon$ moments

The barrier to reaching polylogarithmic separation

Barrier: Clustering with separation $\Omega(k^\epsilon)$ requires degree $1/\epsilon$ moments

To get separation $\mathrm{poly}(\log k)$, this corresponds to taking degree $t = \Theta(\log k / \log\log k)$ moments.

The barrier to reaching polylogarithmic separation

Barrier: Clustering with separation $\Omega(k^{\epsilon})$ requires degree $1/\epsilon$ moments

To get separation $\mathrm{poly}(\log k)$, this corresponds to taking degree $t = \Theta(\log k / \log\log k)$ moments.

Problem

The moment tensor $\mathbb{E}_{X\sim\mathcal{M}}[X^{\otimes t}]$ *requires a quasipolynomial number of samples to estimate accurately.*

The barrier to reaching polylogarithmic separation

Barrier: Clustering with separation $\Omega(k^\epsilon)$ requires degree $1/\epsilon$ moments

To get separation $\mathrm{poly}(\log k)$, this corresponds to taking degree $t = \Theta(\log k / \log \log k)$ moments.

Problem

The moment tensor $\mathbb{E}_{X \sim \mathcal{M}}[X^{\otimes t}]$ *requires a quasipolynomial number of samples to estimate accurately.*

Problem

The moment tensor $\mathbb{E}_{X \sim \mathcal{M}}[X^{\otimes t}]$ *requires quasipolynomial time to write down.*

Outline

- The barrier to existing approaches
- **Our techniques**
  - **Implicit Moment Estimation**
  - Implicit Moment Computation
- Putting it all together

Our Approach

Our Approach

We will still use information about moments of degree $t = \Theta(\log k / \log \log k)$

Our Approach

We will still use information about moments of degree $t = \Theta(\log k / \log\log k)$

We develop new techniques for accessing/manipulating this information more efficiently

Important Ingredients

Important Ingredients

1. Estimating degree $t = \Theta(\log k / \log\log k)$ moments accurately

Important Ingredients

1. Estimating degree $t = \Theta(\log k/\log\log k)$ moments accurately

2. Representing the degree $t = \Theta(\log k/\log\log k)$ moment tensor efficiently

Important Ingredients

1. Estimating degree $t = \Theta(\log k / \log \log k)$ moments accurately
   - We show certain projections of the moment tensor have *polynomially bounded variance*
   - We can estimate these projections *sample-efficiently*

2. Representing the degree $t = \Theta(\log k / \log \log k)$ moment tensor efficiently

Important Ingredients

1. Estimating degree $t = \Theta(\log k / \log \log k)$ moments accurately
   - We show certain projections of the moment tensor have *polynomially bounded variance*
   - We can estimate these projections *sample-efficiently*

2. Representing the degree $t = \Theta(\log k / \log \log k)$ moment tensor efficiently
   - We only need to perform a restricted set of operations on the moment tensor
   - These can be *performed implicitly in polynomial time*

Important Ingredients

1. **Estimating degree $t = \Theta(\log k / \log \log k)$ moments accurately**
   - We show certain projections of the moment tensor have *polynomially bounded variance*
   - We can estimate these projections *sample-efficiently*

2. Representing the degree $t = \Theta(\log k / \log \log k)$ moment tensor efficiently
   - We only need to perform a restricted set of operations on the moment tensor
   - These can be *performed implicitly in polynomial time*

Reducing to the difference mixture

Reducing to the difference mixture

Instead of working directly with the mixture $\mathcal{M} = \sum_{i=1}^{k} \frac{1}{k} N(\mu_i, I)$, we will work with the *difference mixture*

Reducing to the difference mixture

Instead of working directly with the mixture $\mathcal{M} = \sum_{i=1}^{k} \frac{1}{k} N(\mu_i, I)$, we will work with the *difference mixture*

Difference Mixture: distribution of the random variable $Y = (X - X')/\sqrt{2}$, for $X, X' \sim \mathcal{M}$.

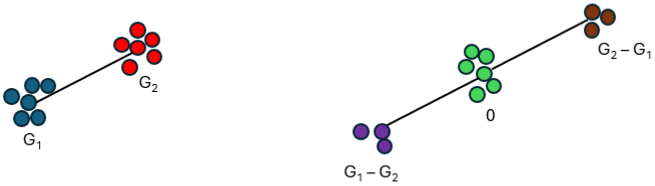
Reducing to the difference mixture

Instead of working directly with the mixture $\mathcal{M} = \sum_{i=1}^{k} \frac{1}{k} N(\mu_i, I)$, we will work with the *difference mixture*

Difference Mixture: distribution of the random variable $Y = (X - X')/\sqrt{2}$, for $X, X' \sim \mathcal{M}$.

This is a new isotropic GMM with $k(k-1)+1$ components

- One component has mean 0
- The rest have mean that is at least Δ-far from 0.



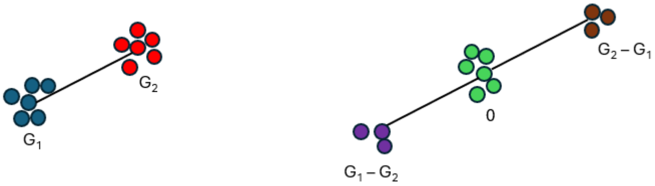
Reducing to the difference mixture

Instead of working directly with the mixture $\mathcal{M} = \sum_{i=1}^{k} \frac{1}{k} N(\mu_i, I)$, we will work with the *difference mixture*

Difference Mixture: distribution of the random variable $Y = (X - X')/\sqrt{2}$, for $X, X' \sim \mathcal{M}$.

This is a new isotropic GMM with $k(k-1)+1$ components

- One component has mean 0
- The rest have mean that is at least Δ-far from 0.



Let $\mathcal{M}$ be the difference mixture for the rest of this talk

Reducing to the difference mixture (cont.)

Reducing to the difference mixture (cont.)

To cluster the original mixture, it suffices to, detect if a sample comes from the 0 component or another component.

Reducing to the difference mixture (cont.)

To cluster the original mixture, it suffices to, detect if a sample comes from the 0 component or another component.

Let $\mathcal{M} = w_0 N(0, I) + \sum_{i=1}^{k} w_i N(\mu_i, I)$ be a difference mixture, so that:

- $w_i \geq 1/\mathrm{poly}(k)$
- $\|\mu_i\|_2 \geq \Delta$

Reducing to the difference mixture (cont.)

To cluster the original mixture, it suffices to, detect if a sample comes from the 0 component or another component.

Let $\mathcal{M} = w_0 N(0, I) + \sum_{i=1}^{k} w_i N(\mu_i, I)$ be a difference mixture, so that:

- $w_i \geq 1/\mathrm{poly}(k)$
- $\|\mu_i\|_2 \geq \Delta$

Problem

Let $X_1, \ldots, X_n$ be a set of polynomially many samples from $\mathcal{M}$. Given a new sample $X' \sim \mathcal{M}$, distinguish between the case where $X' \sim N(0, I)$, and $X' \sim N(\mu_i, I)$, for some $i \geq 1$.

Reducing to the difference mixture (cont.)

To cluster the original mixture, it suffices to, detect if a sample comes from the 0 component or another component.

Let $\mathcal{M} = w_0 N(0, I) + \sum_{i=1}^{k} w_i N(\mu_i, I)$ be a difference mixture, so that:

- $w_i \geq 1/\mathrm{poly}(k)$
- $\|\mu_i\|_2 \geq \Delta$

Problem

Let $X_1, \ldots, X_n$ be a set of polynomially many samples from $\mathcal{M}$. Given a new sample $X' \sim \mathcal{M}$, distinguish between the case where $X' \sim N(0, I)$, and $X' \sim N(\mu_i, I)$, for some $i \geq 1$.

For simplicity, also assume that $\Delta = \mathrm{poly}(\log k)$, and $\|\mu_i\|_2 = \mathrm{poly}(\log k)$, for all i.

Test Functions

Test Functions

Goal: design a test function for distinguishing

- Given a sample $X \sim \mathcal{M}$, we compute the test function $f(X)$

Test Functions

Goal: design a test function for distinguishing

- Given a sample $X \sim \mathcal{M}$, we compute the test function $f(X)$

We want the following properties:

1. $f(X)$ is small with high probability if X is from the 0-component
2. $f(X)$ is large with high probability if X is from a component with mean bounded away from 0

Test Functions

Goal: design a test function for distinguishing

- Given a sample $X \sim \mathcal{M}$, we compute the test function $f(X)$

We want the following properties:

1. $f(X)$ is small with high probability if X is from the 0-component
2. $f(X)$ is large with high probability if X is from a component with mean bounded away from 0

Test function f will be a polynomial of degree t. The key will be to bound the variance.

The Hermite polynomial tensor

The Hermite polynomial tensor

- In 1D, Hermite polynomials are $h_{m+1}(x) = xh_m(x) - mh_{m-1}(x)$

The Hermite polynomial tensor

- In 1D, Hermite polynomials are $h_{m+1}(x) = xh_m(x) - mh_{m-1}(x)$
- $h_1(x) = x, h_2(x) = x^2 - 1, h_3(x) = x^3 - 3x, \ldots$

The Hermite polynomial tensor

- In 1D, Hermite polynomials are $h_{m+1}(x) = xh_m(x) - mh_{m-1}(x)$
- $h_1(x) = x, h_2(x) = x^2 - 1, h_3(x) = x^3 - 3x, \ldots$
- **Key property:** $\mathbb{E}_{x \sim N(\mu,1)}[h_t(x)] = \mu^t$

The Hermite polynomial tensor

- In 1D, Hermite polynomials are $h_{m+1}(x) = xh_m(x) - mh_{m-1}(x)$
- $h_1(x) = x, h_2(x) = x^2 - 1, h_3(x) = x^3 - 3x, \ldots$
- **Key property:** $\mathbb{E}_{x \sim N(\mu,1)}[h_t(x)] = \mu^t$

- In higher dimensions we can construct an analog h_t where $h_t(X)$ for $X \in \mathbb{R}^d$ is a tensor in $\mathbb{R}^{d^{\otimes t}}$ that is a polynomial in X

The Hermite polynomial tensor

- In 1D, Hermite polynomials are $h_{m+1}(x) = xh_m(x) - mh_{m-1}(x)$
- $h_1(x) = x, h_2(x) = x^2 - 1, h_3(x) = x^3 - 3x, \ldots$
- **Key property:** $\mathbb{E}_{x \sim N(\mu,1)}[h_t(x)] = \mu^t$

- In higher dimensions we can construct an analog h_t where $h_t(X)$ for $X \in \mathbb{R}^d$ is a tensor in $\mathbb{R}^{d^{\otimes t}}$ that is a polynomial in X
- $h_1(x) = X, h_2(x) = X^{\otimes 2} - I_{d \times d}, h_3(X) = X^{\otimes 3} - \sum_{sym} I_{d \times d} \otimes X, \ldots$

The Hermite polynomial tensor

- In 1D, Hermite polynomials are $h_{m+1}(x) = xh_m(x) - mh_{m-1}(x)$
- $h_1(x) = x, h_2(x) = x^2 - 1, h_3(x) = x^3 - 3x, \ldots$
- **Key property:** $\mathbb{E}_{x\sim N(\mu,1)}[h_t(x)] = \mu^t$

- In higher dimensions we can construct an analog h_t where $h_t(X)$ for $X \in \mathbb{R}^d$ is a tensor in $\mathbb{R}^{d^{\otimes t}}$ that is a polynomial in X
- $h_1(x) = X, h_2(x) = X^{\otimes 2} - I_{d\times d}, h_3(X) = X^{\otimes 3} - \sum_{sym} I_{d\times d} \otimes X, \ldots$
- **Key property:** $\mathbb{E}_{X\sim N(\mu,I)}[h_t(X)] = \mu^{\otimes t}$

Properties of the Hermite polynomial tensor

Properties of the Hermite polynomial tensor

- $h_t(X)$ is an unbiased estimator for $\mu^{\otimes t}$

Properties of the Hermite polynomial tensor

- $h_t(X)$ is an unbiased estimator for $\mu^{\otimes t}$
- It has bounded variance, i.e. for any $v \in \mathbb{R}^{d^t}$ with $\|v\| = 1$,

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X) \rangle^2 \right] \leq O(t)^t = \mathrm{poly}(k) .$$

Properties of the Hermite polynomial tensor

- $h_t(X)$ is an unbiased estimator for $\mu^{\otimes t}$
- It has bounded variance, i.e. for any $v \in \mathbb{R}^{d^t}$ with $\|v\| = 1$,

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X) \rangle^2 \right] \leq O(t)^t = \mathrm{poly}(k) \; .$$

- It reliably witnesses large means, i.e. if $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then with high probability,

$$\langle h_t(X), \mu^{\otimes t} \rangle \geq (0.8 \, \|\mu\|)^{2t} \geq \mathrm{poly}(k) \; .$$

Initial Attempt

Initial Attempt

Properties of the Hermite Polynomial Tensor

- **Bounded Variance:** for all unit vectors $v \in \mathbb{R}^{d^t}$

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X) \rangle^2 \right] \leq \mathrm{poly}(k) .$$

- **Large Signal:** If $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then w.h.p.

$$\langle h_t(X), \mu^{\otimes t} \rangle \geq (0.8 \|\mu\|)^{2t} \geq \mathrm{poly}(k) .$$

Initial Attempt

Properties of the Hermite Polynomial Tensor

- **Bounded Variance:** for all unit vectors $v \in \mathbb{R}^{d^t}$

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X)\rangle^2\right] \leq \mathrm{poly}(k)\,.$$

- **Large Signal:** If $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then w.h.p.

$$\langle h_t(X), \mu^{\otimes t}\rangle \geq (0.8\,\|\mu\|)^{2t} \geq \mathrm{poly}(k)\ .$$

Attempt: Try $f(X) = \|h_t(X)\|$ i.e. check whether $\|h_t(X)\|$ is sufficiently large

Initial Attempt

Properties of the Hermite Polynomial Tensor

- **Bounded Variance:** for all unit vectors $v \in \mathbb{R}^{d^t}$

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X) \rangle^2 \right] \leq \mathrm{poly}(k) \,.$$

- **Large Signal:** If $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then w.h.p.

$$\langle h_t(X), \mu^{\otimes t} \rangle \geq (0.8 \, \|\mu\|)^{2t} \geq \mathrm{poly}(k) \,.$$

Attempt: Try $f(X) = \|h_t(X)\|$ i.e. check whether $\|h_t(X)\|$ is sufficiently large

Issue: we only know that the variance of $h_t(X)$ in each direction is bounded but it has d^t entries which is too many

Projecting onto the "Signal" Subspace

Projecting onto the "Signal" Subspace

Properties of the Hermite Polynomial Tensor

- **Bounded Variance:** for all unit vectors $v \in \mathbb{R}^{d^t}$

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X) \rangle^2 \right] \leq \mathrm{poly}(k) \,.$$

- **Large Signal:** If $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then w.h.p.

$$\langle h_t(X), \mu^{\otimes t} \rangle \geq (0.8 \, \|\mu\|)^{2t} \geq \mathrm{poly}(k) \,.$$

Projecting onto the "Signal" Subspace

Properties of the Hermite Polynomial Tensor

- **Bounded Variance:** for all unit vectors $v \in \mathbb{R}^{d^t}$

$$\mathop{\mathbb{E}}_{X\sim N(0,I)} \left[\langle v, h_t(X)\rangle^2\right] \leq \mathrm{poly}(k)\,.$$

- **Large Signal:** If $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then w.h.p.

$$\langle h_t(X), \mu^{\otimes t}\rangle \geq (0.8\,\|\mu\|)^{2t} \geq \mathrm{poly}(k)\,.$$

Main Idea: We instead let $f(X) = \|\Pi h_t(X)\|$ where Π projects onto a low dimensional subspace that "captures the signal"

Projecting onto the "Signal" Subspace

Properties of the Hermite Polynomial Tensor

- **Bounded Variance:** for all unit vectors $v \in \mathbb{R}^{d^t}$

$$\mathop{\mathbb{E}}_{X \sim N(0,I)} \left[\langle v, h_t(X)\rangle^2\right] \leq \mathrm{poly}(k)\,.$$

- **Large Signal:** If $\|\mu\| \geq \Omega(t^{1/2})$ and $X \sim N(\mu, I)$, then w.h.p.

$$\langle h_t(X), \mu^{\otimes t}\rangle \geq (0.8\,\|\mu\|)^{2t} \geq \mathrm{poly}(k)\,.$$

Main Idea: We instead let $f(X) = \|\Pi h_t(X)\|$ where Π projects onto a low dimensional subspace that "captures the signal"

Want Π to project onto $\mathrm{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$ - which is k-dimensional!

Designing the Test Functions

Designing the Test Functions

Test Function: set $f(x) = \|\Pi_t h_t(X)\|$ where Π_t to projects onto $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Designing the Test Functions

Test Function: set $f(x) = \|\Pi_t h_t(X)\|$ where Π_t to projects onto $\mathrm{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Recall we need to verify the following properties

1. $f(X)$ is small with high probability if X is from the 0-component

2. $f(X)$ is large with high probability if X is from a component with mean bounded away from 0

Designing the Test Functions

Test Function: set $f(x) = \|\Pi_t h_t(X)\|$ where Π_t to projects onto $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Recall we need to verify the following properties

1. $f(X)$ is small with high probability if X is from the 0-component

Lemma: 0-component

Let $X \sim N(0, I)$. Then $\|\Pi_t h_t(X)\| \leq k^{1/2} \cdot O(t)^{t/2}$ with high probability.

2. $f(X)$ is large with high probability if X is from a component with mean bounded away from 0

Designing the Test Functions

Test Function: set $f(x) = \|\Pi_t h_t(X)\|$ where Π_t to projects onto $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Recall we need to verify the following properties

1. $f(X)$ is small with high probability if X is from the 0-component

Lemma: 0-component

Let $X \sim N(0, I)$. Then $\|\Pi_t h_t(X)\| \leq k^{1/2} \cdot O(t)^{t/2}$ with high probability.

2. $f(X)$ is large with high probability if X is from a component with mean bounded away from 0

Lemma: Nonzero-component

Let $X \sim N(\mu_i, I)$, where $\|\mu_i\| \geq \Omega(t^{1/2})$. Then $\|\Pi_t h_t(X)\| \geq (0.8 \|\mu_i\|)^t$ with high probability.

Verifying Soundness

Verifying Soundness

First, for simplicity, assume we exactly know $\Pi_t = \text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Verifying Soundness

First, for simplicity, assume we exactly know $\Pi_t = \text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

If $\Delta \geq \Omega\left(\log^{1/2+c} k\right)$, and $t = \Theta\left(\frac{\log k}{\log \log k}\right)$, then

$$\underbrace{k^{1/2} \cdot O(t)^{t/2}}_{\text{Zero Case}} \ll \underbrace{(0.8\,\|\mu_i\|)^t}_{\text{Nonzero Case}} .$$

Verifying Soundness

First, for simplicity, assume we exactly know $\Pi_t = \text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

If $\Delta \geq \Omega\left(\log^{1/2+c} k\right)$, and $t = \Theta\left(\frac{\log k}{\log \log k}\right)$, then

$$\underbrace{k^{1/2} \cdot O(t)^{t/2}}_{\text{Zero Case}} \ll \underbrace{(0.8 \|\mu_i\|)^t}_{\text{Nonzero Case}}.$$

This gives us a way to solve the distinguishing problem!

Verifying Soundness

First, for simplicity, assume we exactly know $\Pi_t = \text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

If $\Delta \geq \Omega\left(\log^{1/2+c} k\right)$, and $t = \Theta\left(\frac{\log k}{\log \log k}\right)$, then

$$\underbrace{k^{1/2} \cdot O(t)^{t/2}}_{\text{Zero Case}} \ll \underbrace{(0.8 \|\mu_i\|)^t}_{\text{Nonzero Case}} .$$

This gives us a way to solve the distinguishing problem!

Takeaway: if we know $\Pi_t = \text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$ then we can cluster with polynomially many samples

Outline

- The barrier to existing approaches
- **Our techniques**
  - Implicit Moment Estimation
  - **Implicit Moment Computation**
- Putting it all together

Important Ingredients

1. Estimating degree $t = \Theta(\log k / \log \log k)$ moments accurately
   - We show certain projections of the moment tensor have *polynomially bounded variance*
   - We can estimate these projections *sample-efficiently*

2. **Representing the degree $t = \Theta(\log k / \log \log k)$ moment tensor efficiently**
   - We only need to perform a restricted set of operations on the moment tensor
   - These can be *performed implicitly in polynomial time*

What Do We Need to Compute?

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Evaluating the Projection: need to compute $\Pi_t h_t(X)$ i.e. apply the projection to a Hermite polynomial tensor

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\mathrm{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Evaluating the Projection: need to compute $\Pi_t h_t(X)$ i.e. apply the projection to a Hermite polynomial tensor

Preview

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\text{span}(\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t})$

Evaluating the Projection: need to compute $\Pi_t h_t(X)$ i.e. apply the projection to a Hermite polynomial tensor

Preview

- **Main idea:** we construct such a representation inductively (in t)

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\text{span}(\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t})$

Evaluating the Projection: need to compute $\Pi_t h_t(X)$ i.e. apply the projection to a Hermite polynomial tensor

Preview

- **Main idea:** we construct such a representation inductively (in t)
- $\Pi_t : \mathbb{R}^{d^t} \to \mathbb{R}^k$ is too large to write down – we will compute an implicit representation of Π_t that has polynomial size and allows us to perform certain restricted operations in polynomial time

Iterative projection maps

Iterative projection maps

Inductive Step: Let $\Pi_{s-1} = \text{span}\left(\mu_1^{\otimes(s-1)}, \ldots, \mu_k^{\otimes(s-1)}\right)$

- Assume we have some implicit representation of Π_{s-1}

Iterative projection maps

Inductive Step: Let $\Pi_{s-1} = \text{span}\left(\mu_1^{\otimes(s-1)}, \ldots, \mu_k^{\otimes(s-1)}\right)$

- Assume we have some implicit representation of Π_{s-1}

Goal: Construct a representation of $\Pi_s = \text{span}\left(\mu_1^{\otimes s}, \ldots, \mu_k^{\otimes s}\right)$

Constructing the Projection (cont.)

Constructing the Projection (cont.)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, estimate

$$\frac{1}{n}\sum_{i=1}^{n} h_{2s}(X_i) \approx \mathop{\mathbb{E}}_{X\sim\mathcal{M}}[h_{2s}(X)] = \sum_{i=1}^{k} w_i \mu_i^{\otimes 2s} .$$

Constructing the Projection (cont.)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, estimate

$$\frac{1}{n}\sum_{i=1}^{n} h_{2s}(X_i) \approx \mathop{\mathbb{E}}_{X\sim\mathcal{M}}[h_{2s}(X)] = \sum_{i=1}^{k} w_i \mu_i^{\otimes 2s} .$$

If we treat this as a $d^s \times d^s$ matrix, we can write this as

$$T_{2s} = \sum_{i=1}^{k} w_i \left(\mu_i^{\otimes s}\right)\left(\mu_i^{\otimes s}\right)^{\top} .$$

Constructing the Projection (cont.)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, estimate

$$\frac{1}{n}\sum_{i=1}^{n} h_{2s}(X_i) \approx \mathop{\mathbb{E}}_{X\sim\mathcal{M}}[h_{2s}(X)] = \sum_{i=1}^{k} w_i \mu_i^{\otimes 2s} \, .$$

If we treat this as a $d^s \times d^s$ matrix, we can write this as

$$T_{2s} = \sum_{i=1}^{k} w_i \left(\mu_i^{\otimes s}\right)\left(\mu_i^{\otimes s}\right)^{\top} \, .$$

This is a rank-k matrix whose nontrivial eigenvectors are exactly the span of $\{\mu_i^{\otimes s}\}$.

Constructing the Projection (cont.)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, estimate

$$\frac{1}{n}\sum_{i=1}^{n} h_{2s}(X_i) \approx \mathop{\mathbb{E}}_{X\sim\mathcal{M}}[h_{2s}(X)] = \sum_{i=1}^{k} w_i \mu_i^{\otimes 2s} \ .$$

If we treat this as a $d^s \times d^s$ matrix, we can write this as

$$T_{2s} = \sum_{i=1}^{k} w_i \left(\mu_i^{\otimes s}\right)\left(\mu_i^{\otimes s}\right)^{\top} \ .$$

This is a rank-k matrix whose nontrivial eigenvectors are exactly the span of $\{\mu_i^{\otimes s}\}$.

This matrix is too large to work with, but we can make use of the inductive step

Constructing the Projection (cont.)

Constructing the Projection (cont.)

Define the projection matrix

$$B_s = I_{d \times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} \ .$$

Constructing the Projection (cont.)

Define the projection matrix

$$B_s = I_{d\times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} .$$

We can instead estimate $A_s \in \mathbb{R}^{dk\times dk}$ given by

$$A_s = \frac{1}{n}\sum_{i=1}^{n} B_s h_{2s}(X_i) B_s^\top \approx \sum_{i=1}^{k} w_i \left(B_s \mu_i^{\otimes s}\right)\left(B_s \mu_i^{\otimes s}\right)^\top$$

Constructing the Projection (cont.)

Define the projection matrix

$$B_s = I_{d\times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} .$$

We can instead estimate $A_s \in \mathbb{R}^{dk\times dk}$ given by

$$A_s = \frac{1}{n}\sum_{i=1}^{n} B_s h_{2s}(X_i) B_s^\top \approx \sum_{i=1}^{k} w_i \left(B_s \mu_i^{\otimes s}\right)\left(B_s \mu_i^{\otimes s}\right)^\top$$

Now we let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ denote the projection onto the top k eigenvectors of A_s

- This approximates the span of $\{B_s \mu_i^{\otimes s}\}$

Constructing the Projection (cont.)

Define the projection matrix

$$B_s = I_{d \times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} .$$

We can instead estimate $A_s \in \mathbb{R}^{dk \times dk}$ given by

$$A_s = \frac{1}{n} \sum_{i=1}^{n} B_s h_{2s}(X_i) B_s^\top \approx \sum_{i=1}^{k} w_i \left(B_s \mu_i^{\otimes s}\right) \left(B_s \mu_i^{\otimes s}\right)^\top$$

Now we let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ denote the projection onto the top k eigenvectors of A_s

- This approximates the span of $\{B_s \mu_i^{\otimes s}\}$

We set $\Pi_s = \Gamma_s B_s$

Analysis without Noise

Analysis without Noise

Define the matrix

$$B_s = I_{d \times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} .$$

and assume $\Pi_{s-1} = \text{span}\left(\mu_1^{\otimes(s-1)}, \ldots, \mu_k^{\otimes(s-1)}\right)$

Analysis without Noise

Define the matrix

$$B_s = I_{d \times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} \ .$$

and assume $\Pi_{s-1} = \operatorname{span}\left(\mu_1^{\otimes(s-1)}, \ldots, \mu_k^{\otimes(s-1)}\right)$

Let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ denote the projection onto the span of $\{B_s \mu_i^{\otimes s}\}$.

Analysis without Noise

Define the matrix

$$B_s = I_{d \times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} \ .$$

and assume $\Pi_{s-1} = \text{span}\left(\mu_1^{\otimes(s-1)}, \ldots, \mu_k^{\otimes(s-1)}\right)$

Let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ denote the projection onto the span of $\{B_s \mu_i^{\otimes s}\}$.

Claim: $\text{span}\left(\mu_1^{\otimes s}, \ldots, \mu_k^{\otimes s}\right) = \Gamma_s B_s$.

Analysis without Noise

Define the matrix

$$B_s = I_{d \times d} \otimes \Pi_{s-1} : \mathbb{R}^{d^s} \to \mathbb{R}^{dk} \ .$$

and assume $\Pi_{s-1} = \text{span}\left(\mu_1^{\otimes(s-1)}, \ldots, \mu_k^{\otimes(s-1)}\right)$

Let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ denote the projection onto the span of $\{B_s \mu_i^{\otimes s}\}$.

Claim: $\text{span}\left(\mu_1^{\otimes s}, \ldots, \mu_k^{\otimes s}\right) = \Gamma_s B_s$.
Proof: It suffices to check that $\Gamma_s B_s$ preserves the norm of all $\mu_i^{\otimes s}$.

$$\begin{aligned} \left\| \Gamma_s B_s \mu_i^{\otimes s} \right\| &= \left\| B_s \mu_i^{\otimes s} \right\| \\ &= \left\| \mu_i \otimes \Pi_{s-1} \mu_i^{\otimes(s-1)} \right\| \\ &= \| \mu_i \| \left\| \Pi_{s-1} \mu_i^{\otimes(s-1)} \right\| = \| \mu_i \|^s \ . \end{aligned}$$

Summary of the Full Construction

Summary of the Full Construction

Given an efficient representation of Π_{s-1}, and samples $X_1, \ldots, X_n \sim \mathcal{M}$:

Summary of the Full Construction

Given an efficient representation of Π_{s-1}, and samples $X_1, \ldots, X_n \sim \mathcal{M}$:

1. Let $B_s = I_{d \times d} \otimes \Pi_{s-1}$.

Summary of the Full Construction

Given an efficient representation of Π_{s-1}, and samples $X_1, \ldots, X_n \sim \mathcal{M}$:

1. Let $B_s = I_{d \times d} \otimes \Pi_{s-1}$.
2. Using samples, estimate the matrix $A_s \in \mathbb{R}^{dk \times dk}$

$$A_s = \frac{1}{n} \sum_{i=1}^{n} B_s h_{2s}(X_i) B_s^\top \approx \sum_{i=1}^{k} w_i \left(B_s \mu_i^{\otimes s}\right) \left(B_s \mu_i^{\otimes s}\right)^\top$$

Summary of the Full Construction

Given an efficient representation of Π_{s-1}, and samples $X_1, \ldots, X_n \sim \mathcal{M}$:

1. Let $B_s = I_{d \times d} \otimes \Pi_{s-1}$.
2. Using samples, estimate the matrix $A_s \in \mathbb{R}^{dk \times dk}$

$$A_s = \frac{1}{n} \sum_{i=1}^{n} B_s h_{2s}(X_i) B_s^\top \approx \sum_{i=1}^{k} w_i \left(B_s \mu_i^{\otimes s}\right) \left(B_s \mu_i^{\otimes s}\right)^\top$$

3. Let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ project onto the top k eigenvectors of A_s.

Summary of the Full Construction

Given an efficient representation of Π_{s-1}, and samples $X_1, \ldots, X_n \sim \mathcal{M}$:

1. Let $B_s = I_{d\times d} \otimes \Pi_{s-1}$.
2. Using samples, estimate the matrix $A_s \in \mathbb{R}^{dk\times dk}$

$$A_s = \frac{1}{n}\sum_{i=1}^{n} B_s h_{2s}(X_i) B_s^\top \approx \sum_{i=1}^{k} w_i \left(B_s \mu_i^{\otimes s}\right)\left(B_s \mu_i^{\otimes s}\right)^\top$$

3. Let $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ project onto the top k eigenvectors of A_s.
4. Output $\Pi_s = \Gamma_s B_s$.

Evaluations with the Implicit Projection

Evaluations with the Implicit Projection

We constructed $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d \times d} \otimes \Pi_{s-1} \right) \ .$$

Evaluations with the Implicit Projection

We constructed $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d \times d} \otimes \Pi_{s-1} \right) \; .$$

Unraveling the recursion, this yields a series of projection matrices $\Gamma_1, \ldots, \Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d \times d} \otimes \Gamma_{s-1} \left(I_{d \times d} \otimes \ldots \right) \right) \; .$$

This is a polynomial-sized implicit representation!

Evaluations with the Implicit Projection

We constructed $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d\times d} \otimes \Pi_{s-1} \right) \ .$$

Unraveling the recursion, this yields a series of projection matrices $\Gamma_1, \dots, \Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d\times d} \otimes \Gamma_{s-1} \left(I_{d\times d} \otimes \dots \right) \right) \ .$$

This is a polynomial-sized implicit representation!

Key Fact: $\Pi_s v$ can be computed efficiently on rank-1 tensors i.e. of the form $v = v_1 \otimes \cdots \otimes v_s$ because

$$\Pi_s(v_1 \otimes \cdots \otimes v_s) = \Gamma_s(v_1 \otimes \Pi_{s-1}(v_2 \otimes \cdots \otimes v_s)$$

Evaluations with the Implicit Projection

We constructed $\Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d \times d} \otimes \Pi_{s-1} \right) .$$

Unraveling the recursion, this yields a series of projection matrices $\Gamma_1, \ldots, \Gamma_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ so that

$$\Pi_s = \Gamma_s \left(I_{d \times d} \otimes \Gamma_{s-1} \left(I_{d \times d} \otimes \ldots \right) \right) .$$

This is a polynomial-sized implicit representation!

Key Fact: $\Pi_s v$ can be computed efficiently on rank-1 tensors i.e. of the form $v = v_1 \otimes \cdots \otimes v_s$ because

$$\Pi_s(v_1 \otimes \cdots \otimes v_s) = \Gamma_s(v_1 \otimes \Pi_{s-1}(v_2 \otimes \cdots \otimes v_s)$$

We can (only) efficiently apply the projection to rank-1 tensors

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Evaluating the Projection: need to compute $\Pi_t h_t(X)$ i.e. apply the projection to a Hermite polynomial tensor

What Do We Need to Compute?

Computing the Projection: need to compute Π_t that projects onto the subspace $\text{span}(\mu_1^{\otimes t}, \ldots, \mu_k^{\otimes t})$

Evaluating the Projection: need to compute $\Pi_t h_t(X)$ i.e. apply the projection to a Hermite polynomial tensor

Want to show: To evaluate $\Pi_t h_t(X)$, we need to represent $h_t(x)$ as a low-rank tensor!

Low rank approximations of Hermite polynomials

Low rank approximations of Hermite polynomials

Unfortunately $h_t(X)$ does not appear to be a low-rank tensor - it contains terms of the form $I_{d \times d} \otimes \cdots \otimes I_{d \times d}$

Low rank approximations of Hermite polynomials

Unfortunately $h_t(X)$ does not appear to be a low-rank tensor - it contains terms of the form $I_{d\times d} \otimes \cdots \otimes I_{d\times d}$

However, we can introduce additional variables $z_1, \ldots, z_t \sim N(0, I)$ and a polynomial R_t such that

$$\mathop{\mathbb{E}}_{z_1,\ldots,z_t\sim N(0,I)}[R_t(X, z_1, \ldots, z_t)] = h_t(X)$$

Low rank approximations of Hermite polynomials

Unfortunately $h_t(X)$ does not appear to be a low-rank tensor - it contains terms of the form $I_{d \times d} \otimes \cdots \otimes I_{d \times d}$

However, we can introduce additional variables $z_1, \ldots, z_t \sim N(0, I)$ and a polynomial R_t such that

$$\mathop{\mathbb{E}}_{z_1, \ldots, z_t \sim N(0,I)} [R_t(X, z_1, \ldots, z_t)] = h_t(X)$$

We view $R_t(X)$ as a polynomial with random coefficients

Low rank approximations of Hermite polynomials (cont.)

Lemma

For all t, there is a (random) polynomial $R_t : \mathbb{R}^d \to \mathbb{R}^{d^t}$ satisfying:

- **Unbiased:** *For all $X \in \mathbb{R}^d$, we have*

$$\mathop{\mathbb{E}}_{R_t}[R_t(X)] = h_t(X) \ .$$

- **Bounded Variance:** *For all $v \in \mathbb{R}^{d^t}$ with $\|v\| = 1$, we have*

$$\mathop{\mathbb{E}}_{X \sim N(\mu, I), R_t}\left[\langle v, R_t(X)\rangle^2\right] \leq O(t)^t \cdot (\|\mu\|^{2t} + 1) \ .$$

- **Low Rank:** *R_t can always be written as a sum of $\mathrm{poly}(k)$ many (explicit) rank-1 tensors.*

Low rank approximations of Hermite polynomials (cont.)

Lemma

For all t, there is a (random) polynomial $R_t : \mathbb{R}^d \to \mathbb{R}^{d^t}$ satisfying:

- **Unbiased:** *For all $X \in \mathbb{R}^d$, we have*

$$\mathbb{E}_{R_t}[R_t(X)] = h_t(X) .$$

- **Bounded Variance:** *For all $v \in \mathbb{R}^{d^t}$ with $\|v\| = 1$, we have*

$$\mathbb{E}_{X \sim N(\mu, I), R_t}\left[\langle v, R_t(X) \rangle^2\right] \leq O(t)^t \cdot (\|\mu\|^{2t} + 1) .$$

- **Low Rank:** *R_t can always be written as a sum of $\mathrm{poly}(k)$ many (explicit) rank-1 tensors.*

Proof: See paper...

Outline

- The barrier to existing approaches
- Our techniques
  - Implicit Moment Estimation
  - Implicit Moment Computation
- **Putting it all together**

The full algorithm (sort of)

The full algorithm (sort of)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, and another sample $X' \sim \mathcal{M}$:

The full algorithm (sort of)

Given samples $X_1, \dots, X_n \sim \mathcal{M}$, and another sample $X' \sim \mathcal{M}$:

1. Let $t = \Theta(\log k / \log\log k)$.

The full algorithm (sort of)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, and another sample $X' \sim \mathcal{M}$:

1. Let $t = \Theta(\log k / \log\log k)$.

2. **Compute Implicit Projection:** Using $X_1, \ldots, X_n$, apply the previous subroutine to find a representation of Π_t as a sequence of projection matrices $\Gamma_1, \ldots, \Gamma_t$.

The full algorithm (sort of)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, and another sample $X' \sim \mathcal{M}$:

1. Let $t = \Theta(\log k / \log \log k)$.
2. **Compute Implicit Projection:** Using $X_1, \ldots, X_n$, apply the previous subroutine to find a representation of Π_t as a sequence of projection matrices $\Gamma_1, \ldots, \Gamma_t$.
3. **Test Sample:** Compute $\alpha = \|\Pi_t R_t(X')\| \approx \|\Pi_t h_t(X')\|$ (computed efficiently using low-rank representation of R_t)

The full algorithm (sort of)

Given samples $X_1, \ldots, X_n \sim \mathcal{M}$, and another sample $X' \sim \mathcal{M}$:

1. Let $t = \Theta(\log k / \log \log k)$.
2. **Compute Implicit Projection:** Using $X_1, \ldots, X_n$, apply the previous subroutine to find a representation of Π_t as a sequence of projection matrices $\Gamma_1, \ldots, \Gamma_t$.
3. **Test Sample:** Compute $\alpha = \|\Pi_t R_t(X')\| \approx \|\Pi_t h_t(X')\|$ (computed efficiently using low-rank representation of R_t)
4. If $\alpha < k^{1/2} O(t)^{t/2}$, say that X' belongs to the 0 mean cluster, otherwise, say it belongs to a non-zero mean cluster.

Generalizing to Poincaré

Generalizing to Poincaré

A distribution is σ-Poincaré if

$$\mathrm{Var}[f(X)] \leq \sigma^2 \, \mathbb{E}\left[\|\nabla f(X)\|^2\right] \; .$$

Well studied class of distributions, including Gaussians, product distributions, and log-concave distributions (thanks to KLS).

Generalizing to Poincaré

A distribution is σ-Poincaré if

$$\mathrm{Var}[f(X)] \leq \sigma^2 \, \mathbb{E}\left[\|\nabla f(X)\|^2\right] .$$

Well studied class of distributions, including Gaussians, product distributions, and log-concave distributions (thanks to KLS).

Our techniques generalize almost directly to Poincaré distributions, by using *adjusted* polynomials in place of Hermite polynomials.

Generalizing to Poincaré

A distribution is σ-Poincaré if

$$\mathrm{Var}[f(X)] \leq \sigma^2 \, \mathbb{E}\left[\|\nabla f(X)\|^2\right] \, .$$

Well studied class of distributions, including Gaussians, product distributions, and log-concave distributions (thanks to KLS).

Our techniques generalize almost directly to Poincaré distributions, by using *adjusted* polynomials in place of Hermite polynomials.

Here, we require separation $\Delta = \Theta(\log^{1+c} k)$, but this is information-theoretically necessary, since Poincaré distributions could have worse concentration.

In Full Generality

In Full Generality

The procedure we discussed will distinguish between the zero-mean cluster and other clusters only if all the nonzero means have comparable norm.

In Full Generality

The procedure we discussed will distinguish between the zero-mean cluster and other clusters only if all the nonzero means have comparable norm.

If the means have drastically different norms, the signals from the smaller means will get “washed out”

In Full Generality

The procedure we discussed will distinguish between the zero-mean cluster and other clusters only if all the nonzero means have comparable norm.

If the means have drastically different norms, the signals from the smaller means will get "washed out"

To circumvent this and obtain a perfect clustering, we exploit the structure of Gaussians to recursively cluster

- It is not clear how to do this recursion for general Poincaré distributions.

Summary

Summary

We give an algorithm for clustering mixtures of isotropic Gaussians with nearly optimal separation

Summary

We give an algorithm for clustering mixtures of isotropic Gaussians with nearly optimal separation

- Overcomes barriers to previous approaches requiring quasi-polynomial time
- Techniques for accessing moment information at degree $t = \Theta(\log k / \log \log k)$ in polynomial time

Summary

We give an algorithm for clustering mixtures of isotropic Gaussians with nearly optimal separation

- Overcomes barriers to previous approaches requiring quasi-polynomial time
- Techniques for accessing moment information at degree $t = \Theta(\log k / \log\log k)$ in polynomial time

More general GMMs/other mixture models?

Summary

We give an algorithm for clustering mixtures of isotropic Gaussians with nearly optimal separation

- Overcomes barriers to previous approaches requiring quasi-polynomial time
- Techniques for accessing moment information at degree $t = \Theta(\log k / \log\log k)$ in polynomial time

More general GMMs/other mixture models?

Other applications of the implicit moment estimation technique?

Summary

We give an algorithm for clustering mixtures of isotropic Gaussians with nearly optimal separation

- Overcomes barriers to previous approaches requiring quasi-polynomial time
- Techniques for accessing moment information at degree $t = \Theta(\log k / \log \log k)$ in polynomial time

More general GMMs/other mixture models?

Other applications of the implicit moment estimation technique?

Thanks!